A SYSTEM AND METHOD OF EPSILON REMOVAL OF WEIGHTED AUTOMATA AND TRANSDUCERS

RELATED APPLICATION

The present application is related to Attorney Docket No. 2001-0226-A entitled "A System and Method of Epsilon Removal of Weighted Automata and Transducers", by Mehryar Mohri, assigned to the assignee of the present invention. The contents of that application are incorporated herein.

1. Field of the Invention

The present invention is directed to a system and method for epsilon removal of weighted automata and transducers. More specifically, the present invention relates to a system and method of computing a weighted automaton B with no epsilons that is equivalent to the input automaton A.

2. Brief Description of the Related Art

In natural language processing and speech recognition systems, certain algorithms and processes have been developed to receive speech signals and determine what word or phrase has been spoken. The algorithms and processes for speech processing applications are a particular application of finite state machines (FSMs). Finite state machines are devices that allow for simple and accurate design of sequential logic and control functions. Finite state machines have many applications and while the present disclosure specifically relates to speech recognition systems, the basic principles disclosed herein may have other applications beyond speech recognition or signal processing. Some examples of other contexts in which the present invention may apply include text-processing systems such as information

extraction systems, or systems for information retrieval, pattern matching, and computational biology. An FSM is called a "finite" state machine because there are only a limited (finite) number of states. In a simple sense, a door is a finite state machine because it can be in one of two states: open or closed.

A main design tool for FSMs is the state transition diagram or state diagram. The state transition diagram illustrates the relationships between the system states and the events that cause the system to change from one state to the next. Figure 1, discussed below, illustrates an example state transition diagram in the speech recognition process. Disclosed herein are previous methods and systems for converting a particular FSM into a more simple and efficient FSM that maintains the same functionality as the original FSM. The previous methods relate to removing unnecessary transitions from state to state within an FSM.

It is well known that finite state techniques have proven invaluable in a variety of natural language processing applications. A tool useful in natural language processing is the weighted finite state automata and weighted finite state transducer. Such a transducer is described in U.S. patent applications 6,032,111 to Mehryar Mohri, assigned to AT&T Corp., and which contents are incorporated herein by reference. Further background maybe gained from "The Design Principles of a Weighted Finite-State Transducer Library" by Mehryar Mohri, Fernando Pereira and Michael Riley, available on the Internet at: http://citeseer.nj.nec.com/mohri00design.html; and "Weighted Automata in Text and Speech Processing", by Mehryar Mohri, Fernando Pereira, and Michael Riley, found on the Internet at: http://citeseer.nj.nec.com/mohri96weighted.html. The contents of these two publications are incorporated herein by reference for background information.

Figure 1 illustrates a weighted automaton 10 where each circle 12 is a "state" and the arrows between the states 14, 16 represent events causing the change of state. The weights given to the state transitions are positive real numbers representing negative logarithms of probabilities. Weights along the paths are added and when several paths correspond to the same string, the weight of the string is the minimum of the weights of those paths. Transitions 16 represent the ε-weighted transitions discussed herein. The ε-weighted transitions are "empty" strings with no value.

Weighted automata are efficient and convenient devices used in many applications such as text, speech and image processing. The automata obtained in such applications are often the result of various complex operations, some of them introducing the empty string "ε". For the most efficient use of an automaton, it is preferable to remove the ε's of automata since in general they introduce delay when used. An algorithm that constructs an automaton B with no ε's equivalent to an input automaton A with ε's is called an "ε-removal" algorithm.

The relevant art does not present ε-removal of unweighted automata as an independent algorithm deserving a specific study. Instead, the ε-removal process is often mixed with other optimization algorithms such as determinization. This usually makes the presentation of determinization more complex and the underlying ε-removal process obscure. Since ε-removal is not presented as an independent algorithm, it is usually not analyzed and its running time complexity not clearly determined.

The present disclosure will provide discussion and explanation of a finite-state library algorithm for removal of an ε-weight on automata and transducers. The concepts of weights to automata, automatons, and transducers, as well as known

algorithms for ε-removal are discussed. One of ordinary skill in the art will be familiar with the terms waited acceptor, weighted transducer, automata, automaton, and semiring. Background and information on these terms may be found in the references cited above as well as other well-known documentation.

Weight in the automata is a generalization of the notion of automaton: each transition of a weighted automaton is assigned a weight in addition to the usual labels. More formally, a weighted acceptor over finite alphabets and a semiring K is a finite directed graph with nodes representing states and arcs representing transitions in which each transition t is labeled with an input $i(t) \in \Sigma$ and a weight $w(t) \in K$.

Furthermore, each state t has an initial weight and a final weight. In a weighted transducer, each transition t has also an output label $o(t) \in \Delta^*$ where Δ is the transducer's output alphabet. A state q is initial if $\lambda(q) \neq \bar{0}$, and final if $p(q) \neq \bar{0}$. For more information on the mathematical operations of the equations dissclosed herein, see Werner Kuich and Arto Salomaa, Semirings, Automata, Languages, number 5 in EATCS Monographs on Theoretical Computer Science, published by Springer-Verlag, Berlin, Germany, 1986. A person of ordinary skill in the art will understand the algebraic operation and symbols used in the formulas disclosed herein. Therefore, an explanation of each algebraic operator and of the various terms used in the formulas is not provided.

In the design of a weighted finite state transducer library, most algorithms operate on general weighted automata and transducers. The general framework for solving all pairs shortest-paths problems -- closed semirings -- is compatible with the abstract notion of weights commonly used, thus it is preferable to include an efficient version of the generic algorithm of the Floyd-Warshall algorithm in some finite-state machine libraries. The Floyd-Warshall algorithm addresses the desire in finite-state

machine libraries to determine the shortest distance between all nodes and all other nodes in the finite state machine. Determining these shortest distance values for each node to each other node relates to the problem of "All-Pairs-Shortest-Path" which is solved by the Floyd-Warshall algorithm. Using the Floyd-Warshall algorithm code can provide the all pairs shortest distances when the weights between the nodes are real numbers representing, for example, probabilities but also when they are strings or regular expressions. This case is useful to generate efficiently a regular expression equivalent to a given automaton. The Floyd-Warshall algorithm is also useful in the general ε-removal algorithm discussed next.

Composition is the key operation on weighted transducers. The composition algorithm in the weighted case is related to the standard unweighted transducer composition and acceptor intersection algorithms, but in general weighted ε-transitions complicate matters. The input or output label of a transducer transition may be the symbol ε representing a null label. A null input label indicates that no symbol needs to be consumed when traversing the transition, and a null output label indicates that no symbol is output when traversing the transition. Null labels are needed because input and output strings do not always have the same length. For example, a word sequence is much shorter than the corresponding phonetic transcription. Null labels are a convenient way of delaying inputs or outputs, which may have important computation effects. In the weighted finite-state transducers used for example in speech recognition, transitions with null labels may also have a weight.

The presence of null labels makes the composition operation for weighted transducers more delicate than that for unweighted transducers. The problem is illustrated with the composition of the two transducers A and B shown in Figures 2(a) and 2(b). Transducer A has output null transitions, while transducer B has input null

transitions. To help understand how these null transitions interact, refer to derived transducers *A'* and *B'* in Figures 2(c) and 2(d). In transducer *A'*, the output null transitions are labeled ε_2, and the corresponding null moves in *B'* are explicitly marked as self-transitions with input label ε_2. Likewise, the input null transitions of *B* are labeled with ε_1 in *B'*, and the corresponding self-transitions in *A'* have output label ε_1. Any transition in the composition of *A* and *B* has corresponding transition in the composition of *A'* and *B'*, but whereas an output null label in *A* or an input null label in *B* corresponds to staying in the same state on the other transducer, in the composition of *A'* and *B'*, the corresponding transition is made from a pair of transitions with matching *A*-output and *B*-input labels ε_i.

Figure 3 illustrates the pseudocode of a generic epsilon removal algorithm for weighted automata. This ε-removal algorithm is explained further in the article: "The Design Principles of a Weighted Finite-State Transducer Library" incorporated above. Given a weighted automaton M_i, the algorithm returns an equivalents weighted automaton M_o without ε-transitions. *TransM[s]* denotes to set of transitions leaving state *s* in an automaton *M*, *Next(t)* denotes the destination state of transition *t*, *i(t)* denotes its input label, and *w(t)* its weight. Lines 1 and 2 extract from M_i the subautomaton M_e containing all the non-ε transitions. Line 3 applies the general all-pairs shortest distance algorithm CLOSURE to M_ε to derive the ε-closure G_ε. The nested loops starting in lines 4, 5 and 6 iterate over all pair of an ε-closure transition e and a non-ε transition t such that the destination of e is the source of *t*. Line 7 looks in M_o for a transition t' with label *I(t)* from *e*'s source from *t*'s destination if it exists, or creates a new one with eight *O* if it does not. This transition is the result of extending *t* "backwords" with the M_i ε-path represented by ε-closure transition *e*. Its weight,

updated in line 8, is the semiring sum of such extended transitions with a given source destination and label.

In most speech processing applications, the appropriate weight algebra is the tropical semiring. Weights are positive real numbers representing negative logarithms of probabilities. Weights along the path are added; when several paths correspond the same string in the weight of the string is the minimum of the weights of those paths.

As noted before, the computation of the key closure requires the computation of the all pairs shortest distances in M_e. In the case of idempotent semirings such as the tropical semiring, the most efficient algorithm available is Johnson's algorithm, which is based on the algorithms of Dykstra and Bellman-Ford. Details regarding Johnson's algorithm are found in T. Cormen, C. Leiserson and R. Rivest, Introduction to Algorithms, published by the MIT Press, Cambridge, MA, 1992.

For running time, complexity of Johnson's algorithm is $O(|Q|2\ log|Q| + |Q||E|)$ when using Fibonacci heaps, but often the more general but less efficient Floyd-Warshall algorithm is used instead because it supports non-idempotent closed semirings. Further details regarding Fibonacci heaps are found in Introduction to Algorithms referenced above. When M_ε is acyclic, the linear time topological sort algorithm is used which also works with non-idempotent semirings.

Figure 4 illustrates the result of an application of ε-removal to a weighted automata of the tropical semiring of Figure 1. The example shows that the ε-removal algorithm generalizes the classical unweighted algorithm by ensuring that the weight of any string accepted by the automaton is preserved in the ε-free result 20. In Figure 4, the states 22 and transitions 24 of the ε-free automaton 20 preserve the weight of any string accepted by the original automaton 10 shown in Figure 1.

There are inefficiencies within the related use of ε-removal algorithms. For example, the ε-removal process is often presented in a mixture of other optimization algorithms such as determinations. See, e.g., A. V. Aho, R. Sethi, and J. D. Ullman, Compilers, Principles, Techniques and Tools, published by Addison Wesley, Reading, MA, 1986. A generalization of the Floyd-Warshall algorithm is typically used in connection with ε-removal and introduces inefficiencies. First, in the ε-removal algorithm explained above, M_ε was decomposed into strongly connected components, and then the Floyd-Warshall algorithm was applied to each component, and then the acyclic algorithm was applied to the component graph of M_ε to compute the final results. These computational inefficiencies experienced by using the Floyd-Warshall algorithm were usually accepted given that each strongly connected component of M_ε is small relative to M_ε's overall size.

Another disadvantage of obscuring the ε-removal process via its use with other algorithms is the difficulty in an independent analysis of the running-time complexity and efficiency of ε-removal. Therefore, accurate assessment of the efficiencies of such ε-removal processes becomes very difficult.

SUMMARY OF THE INVENTION

What is needed in the art is an independent ε-removal method for weighted automata and transducers defined over a semiring. The algorithm disclosed herein improves the efficiency of weighted automata used in applications such as speech recognition systems, speech synthesis systems and text processing systems. The ε-removal method is preferably used in the case of unweighted automata and

transducers and weighted automata and transducers defined over the tropical semiring based on the general shortest distance algorithm described herein.

An improved ε-removal method is disclosed that computes for any input weighted automaton A with ε-transitions an equivalent weighted automaton B with no ε-transitions. The method comprises two main steps. The first step consists of computing for each state "p" of the input automaton A its ε-closure denoted by C[p]:

$$C[p] = \{(q,w): q \in \varepsilon[p], d[p,q] = w \in K - \{\bar{0}\}\}$$

Where $\varepsilon[p]$ represents a set of states reachable from "p" via a path labeled with ε. The second step consists of modifying the outgoing transitions of each state "p" by removing those labels with ε and by adding to be non-ε-transitions leading each state q with their weights pre-$\otimes$-multiplied by $d[p,q]$.

The second step in the method comprises modifying the outgoing transitions of each state "p" by removing those labeled with ε. The method adds to the set of transitions leaving the state "p" non-ε-transitions leaving each state "q" in the set of states reachable from "p" via a path labeled with ε with their weights pre-$\otimes$-multiplied by the ε-distance from state "p" to state "q" in the automaton A. State "p" is a final state if some state "q" within the set of states reachable from "p" via a path labeled with ε is final and the final weight ρ [p] is

$$\rho[p] = \bigoplus_{q \in e[p] \cap F} (d[p,q] \otimes \rho[q]).$$

The resulting automaton B is the equivalent of automaton A without the ε-transitions. Furthermore, the method disclosed herein has been implemented and is up to 600 times faster than previously known methods based on generalizations of algorithms such as the Floyd-Warshall algorithm.

The epsilon removal algorithm according to an aspect of the present invention of weighted automatic and transducers is a general optimization algorithm useful in all applications where weighted automata and transducers are used: speech recognition, speech synthesis, text processing, genome processing, information extraction, etc. Thus the present invention provides a dramatic improvement in the removal of epsilons from word lattices, which are compact representations of a large number of recognition hypotheses with the associated weights, produced by a recognizer. The invention is preferably applied before the application of other optimizations such as weighted determinization and minimization.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing advantages of the present invention will be apparent from the following detailed description of several embodiments of the invention with reference to the corresponding accompanying drawings, in which:

Figure 1 illustrates a weighted automaton;

Figures 2(a) - 2(b) illustrate the problem of the present of no labels that makes the composition operation for weighted transducers more difficult;

Figures 2(c) - 2(d) illustrate how null transitions interact with relation to two derived transducers A' and B';

Figure 3 illustrates a pseudocode of the related art general ε-removal algorithm;

Figure 4 illustrates a weighted automaton after an ε-removal process;

Figure 5 (a) illustrates a tropical semiring for a weighted automaton A with ε transitions;

Figure 5(b) illustrates a weighted automaton B equivalent to A result of the ε-removal algorithm according to the first embodiment of the present invention;

Figure 5(c) illustrates a weighted automaton C equivalent to A obtained by application of ε-removal to the reverse of A;

Figure 6(a) illustrates ε - removal in the real semiring *(R, +, *, 0, 1)* for a weighted automaton A, where A_ε is k-closed for *(R, +, *, 0, 1)*;

Figure 6(b) illustrates a weighted automaton B equivalent to A output of the ε-removal algorithm;

Figure 7 illustrates a generic single source shortest distance algorithm used in connection with the first step in the first embodiment of present invention;

Figures 8(a) and 8(b) illustrate another aspect of the first embodiment of the invention regarding an input ε-normalization method for weighted transducers; and

Figures 9(a) and 9(b) illustrate another aspect of the first embodiment of the invention regarding an input ε-normalization method for weighted transducers.

DETAILED DESCRIPTION OF THE INVENTION

Presented and disclosed herein is a novel generic ε-removal system and method for weighted automata and transducers defined over a semiring. The system or method can be used with any large class of semirings (framework) that covers the practical case of the semirings currently used for speech processing but that also includes other semirings that can be useful in speech and other domains. The system and method of the present invention also works with any queue discipline adopted, e.g., first-in-first-out (FIFO), shortest-first, etc. The present invention works with any k-closed semiring.

The present invention is particularly useful for unweighted automata and transducers and weighted automata and transducers defined over the tropical semiring. The invention is based on a general shortest-distance algorithm that is described briefly herein.

The method of the present invention is the first embodiment and will be described using example pseudocode and its running time complexity. The present disclosure further discusses the more efficient case of acyclic automata, an on-the-fly implementation of the method and an approximation method in the case of the semirings not covered by a specific framework.

First Embodiment

The first embodiment is illustrated with several semirings. Also described is an input ε-normalization method for weighted transducers, which is based on the general shortest-distance algorithm. The ε-normalization method, which works with all semirings covered by a specific framework, admits an on-the-fly implementation. An on-the-fly implementation allows one to use the algorithm with a lazy evaluation. In other words, instead of removing all the epsilons of the input machine all at once, one can remove only the epsilons that belong to the paths of the input a machine that one might be interested in. So, when some part of the input machine A is read, the output B (w/o epsilons) is constructed just for that part of A.

As discussed above, weighted automata are automata in which the transitions are labeled with weights in addition to the usual alphabet symbols. For various operations to be well defined, the weight set needs to have the algebraic structure of a semiring. What follows includes some preliminary definitions and explanations of some terms necessary for understanding the present invention. A system $K \oplus \otimes \bar{0} \bar{1}$ is a right semiring if:

1. $(K \oplus \bar{0})$ is a commutative monoid with $\bar{0}$ as the identity element for $\oplus$,

2. $(K \otimes \bar{1})$ is a monoid with $\bar{1}$ as the identity element for $\otimes$,

3. $\otimes$ right distributes over $\oplus$: $\forall a,b,c \in K, (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$,

4. $\bar{0}$ is an annihilator for $\otimes$: $\forall a \in K, a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$.

Left semirings are defined in a similar way by replacing right distributivity with left distributivity. $(K, \oplus, \otimes, \bar{0}, \bar{1})$ is a semiring if both left and right distributivity hold. Thus, more informally, a semiring is a ring that may lack negation. As an example, (N, +, ·, 0, 1) is a semiring defined on the set of nonnegative integers N.

A semiring $(K, \oplus, \otimes, \bar{0}, \bar{1})$ is said to be idempotent if for any $a \in K, a \oplus a = a$. The Boolean semiring β = ({0, 1}, V, Λ, 0, 1) and the tropical semiring $T = (RU\{\infty\}, \min, +, \infty, 0)$ are idempotent, but (N, +, · , 0, 1) is not.

As a second definition, a weighted automaton A = (Σ, Q, *I, F, E,* λ *ρ)* over the semiring K is a '7-tuple where Σ is the finite alphabet of the automaton, Q is a finite set of states, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ the set of final states, $E \subseteq Q \times \Sigma \times K \times Q$ a finite set of transitions, $\lambda : I \to K$ is the initial weight function mapping I to K, and $\rho : F \to K$ is the final weight function mapping F to K.

Given a transition $\varepsilon \in E$, according to the present disclosure, *i[e]* is denoted as its input label, *w[e]* its weight, *p[e]* its origin or previous state and *n[e]* its destination state or next state. Given a state $q \in Q$, *E[q]* is the set of transitions leaving q, and by $E^R[q]$ the set of transitions entering q.

A path $\pi = e_1 \ldots e_k$ in A is an element of E^* with consecutive transitions: $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. The variables n and p are extended to paths by setting: $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$. The notation $P(q,q')$ represents the set of paths from q to q'. P can be extended to subsets $R \subseteq Q$ $R' \subseteq Q$, by:

$$P(R,R') = \bigcup_{q \in R,\, q' \in R'} P(q,q')$$

The labeling function i and the weight function w can also be extended to paths by

defining the label of a path as the concatenation of the labels of its constituent transitions, and the weight of a path as the ⊗-product of the weights of its constituent transitions:

$$i[\pi] = i[e_1] \cdots i[e_k]$$

$$w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$$

Given a string $\chi \in \Sigma^*$, $P(\chi)$ represents the set of paths from I to F labeled with χ:

$$P(\chi) = \{\pi \in P(I, F): i[\pi] = \chi\}$$

The output weight associated by A to an input string $\chi \in \Sigma^*$ is:

$$[A](\chi) = \bigoplus_{\pi \in P(x)} \lambda(p[\pi]) \otimes w[\pi] \otimes p(n[\pi])$$

If $P(x) = \bar{0}$, [A] (χ) is defined to be $\bar{0}$. Note that weighted automata over the Boolean semiring are equivalent to the classical unweighted finite automata.

These definitions can be easily generalized to cover the case of any weighted automata with ε-transitions. An ε-removal algorithm computes for any input weighted automaton A with ε-transitions an equivalent weighted automaton B with no ε-transition, that is such that:

$$\forall \chi \in \Sigma^*, [A](\chi) = [B](\chi)$$

Weighted finite-state transducers are defined in a similar way.

As a further definition, a weighted transducer $T = (\Sigma, \Omega, Q, I, F, E, \lambda, \rho)$ over the semiring K is an 8-tuple where Σ is the finite input alphabet of the transducer, Ω is the finite output alphabet of T, Q is a finite set of states, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ the set of final states, $E \subseteq Q \times \Sigma \cup \{oo\} \times \Omega, \cup \{oo\} \times K \times Q$ a finite set of transitions, λ: $I \rightarrow K$ the initial weight function mapping I to K, and ρ: $F \rightarrow K$ the

final weight function mapping F to K.

The output weight associated by T to an input string $x \in \Sigma$ and output string $\gamma \in \Omega$ is:

$$[T](\chi,\gamma) = \bigoplus_{\pi \in P(x,y)} \lambda(p[\pi]) \otimes w[\pi] \otimes p(n[\pi])$$

where *P(x, y)* is the set of paths with input label x and output label y.

The following definitions will help to define the framework for the generic ε-removal algorithm of an embodiment of the present invention. Let $k \geq 0$ be an integer. A commutative semiring $(K \oplus \otimes \bar{0}\,\bar{1})$ is k-closed if:

$$\forall \alpha \in K, \bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$$

When $k = 0$, the previous expression can be rewritten as:

$$\forall \alpha \in K, \bar{1} \oplus \alpha = \bar{1}$$

and K is then said to be bounded. Semirings such as the Boolean semiring β= $(\{0, 1\}, V, \Lambda, 0,1)$ and the tropical semiring $T = (R_+ \cup \{\infty\}, \min, +\infty, 0)$ are bounded.

As yet another definition, let $k \geq 0$ be an integer, let $(K \oplus \otimes \bar{0}\,\bar{1})$ be a commutative semiring, and let A be a weighted automaton over K. $(K \oplus \otimes \bar{0}\,\bar{1})$ is right k-closed for A if for any cycle π of A:

$$\bigoplus_{n=0}^{k+1} \omega[\pi]^n = \bigoplus_{n=0}^{k} \omega[\pi]^n$$

By definition, if K is k-closed, then it is k-closed for any automaton over K.

Given the above definitions and preliminary information, the first aspect of the first embodiment of the present invention is described next. This embodiment generally discloses a method of ε-removal for automatons. A second aspect of the

first embodiment of the invention, discussed below, relates to an input ε-normalization method for transducers. Further embodiments disclosed below relate to an apparatus or system for ε-removal.

According to the first embodiment of the invention, let $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ be a weighted automaton over the semiring K with ε-transitions. We denote by A_ε the automaton obtained from A by removing all transitions not labeled with ε. We present a general ε-removal algorithm for weighted automata based on a generic shortest-distance algorithm that works for any semiring K k-closed for A_ε. It is assumed that K has this property. This class of semirings includes in particular the Boolean semiring $(\{0, 1\}, V, \Lambda, 0, 1)$, the tropical semiring $(R_+ \cup \{\infty\}, \min, +\infty, 0)$ and other semirings not necessarily idempotent.

The method of the first embodiment of the invention is presented by way of example for the case of weighted automata. The case of weighted transducers can be straightforwardly derived from the automata case by viewing a transducer as an automaton over the alphabet $\Sigma \cup \{\varepsilon\} \times \Sigma \cup \{\varepsilon\}$. An ε-transition of a transducer is then a transition labeled with $(\varepsilon, \varepsilon)$.

For p, q in Q, the ε-distance from p to q in the automaton A is denoted by $d[p, q]$ and defined as:

$$d[p,q] = \bigoplus_{\pi \in P(p,q), i[\pi]=\varepsilon} \omega[\pi]$$

This distance is well-defined for any pair of states (p,q) of A when K is a semiring k-closed for A_ε. By definition, for any $p \in Q$, $d[p,p] = \bar{1}$. The value $d[p,q]$ is the distance from p to q in A_ε. The class of semirings with which the generic shortest-distance algorithm works can in fact be extended to that of right semirings

right k-closed for A_ε. Related disclosure is found in U.S. Patent Application No. 09/495174, filed February 1, 2001, by Mehryar Mohri, the contents of which application are incorporated herein.

The first embodiment of the invention comprises two main steps. The first step comprises computing for each state p of the input automaton A its ε-closure denoted by $C[p]$:

$$C[p] = \{(q,w) : q \in \epsilon[p], d[p,q] = w \in K — \{\bar{O}\}\}$$

where $\epsilon[p]$ represents the set of states reachable from p via a path labeled with ϵ. This step is described in more detail later with a discussion of ε-closures. The epsilon-closure step determines for each state p the set of states q that can be reached from p by paths labeled with epsilon. It also gives the total weight corresponding to all the epsilon-paths from p to q. The states q and their weights are used in the second step to remove all the epsilon-paths from p to q, to create new transitions from p to q with their weights adjusted with the weights of the states q.

The second step comprises modifying the outgoing transitions of each state p by removing those labeled with ϵ and by adding to $E[p]$ non-ϵ-transitions leaving each state $q \in \epsilon[p]$ with their weights pre-$\otimes$-multiplied by $d[p, q]$. The following is example ε-removal(A) pseudocode related to the second step of this embodiment of the invention.

```
1   for each p ∈ Q
2     do E[p] ← {e ∈ E[p] : i[e] ≠ ε}
3       for each (q,w) ∈ C[p]
4           do E[p] ← E[p] ∪ {(p,a,w ⊗ w',r) : (q, a, w', r) ∈ E[q],a ≠ ε}
5             if q ∈ F
6                then if p ∉ F
7                   then F ← F ∪ {p}
8                     ρ[p] ← p[p] ⊕ (ω ⊕ ρ[q])
```

State p is a final state if some state $q \in \epsilon[p]$ is final. If so, then the final

weight $\rho[p]$ is then:

$$\rho[p] = \bigoplus_{q \in e[p] \cap F} (d[p,q] \otimes \rho[q])$$

Note the difference in the equations between the Greek letter "ρ" and the English letter "p". After removing ε's at each state p, some states may become inaccessible if they could only be reached by ε-transitions originally. Those states can be removed from the result in time linear in the size of the resulting machine using, for example, a depth-first search of the automaton. A depth-first search of the automaton may be accomplished, for example, by a classic algorithm disclosed in Introduction to Algorithm, by T. Cormen, C. Leiserson and R. Rivest, published by the MIT Press, Cambridge, MA, 1992.

Figures 5(a)-(c) illustrate the use of the algorithm in the specific case of the tropical semiring. A tropical semiring A 30 is shown in Figure 5(a). The states 32 are connected with some non-ε-transitions 34 and some ε-transitions 36. The ε-transitions 36 can be removed in at least two ways: in the way described above, or the reverse.

Figure 5(b) represents an automaton B produced by the ε-removal method of the first embodiment of the invention. Each state 42 is connected by non-ε-transitions 44 (only one transition is labeled but they are all non-ε-transitions). Figure 5(c) represents an automaton C produced by applying ε-removal to the reverse of the automaton and re-reversing the result. As illustrated by Figures 5(b) and 5(c), different results can be obtained when applying the ε-removal method to an automaton or the reverse of an automaton. The different result is due to two factors that are independent of the semiring.

First, the number of states in the original automaton A whose incoming transitions (or outgoing transitions in the reverse case) are all labeled with the empty

string. As mentioned before, those states can be removed from the result. For example, state 3 of the automaton of Figure 5 (a) can only be reached by ε-transitions and admits only outgoing transitions labeled with ε. Thus, that state does not appear in the result in both methods (Figures 5(b)-(c)). The incoming transitions of state 2 are all labeled with ε and thus it does not appear in the result of the ε-removal with the first method, but it does in the reverse method because the outgoing transitions of state 2 are not all labeled with ε.

The second factor is that the total number of non ε-transitions of the states that can be reached from each state q in A_ε (the reverse of A_ε, in the reverse case). This corresponds to the number of outgoing transitions of q in the result of ε-removal.

In practice, heuristics may reduce the number of states and transitions of the resulting machine although this will not affect the worst-case complexity of the method disclosed herein. One can for instance remove some ε-transitions in the reverse way when that creates less transitions and others in the way corresponding to the first method when that helps reducing the resulting size.

Figures 6(a)-(b) illustrate the application of the ε-removal algorithm of the first embodiment of the invention in the case of another semiring, the semiring of real numbers, or automaton A 60. As shown in Figure 6(a), the states 62 are connected via non-ε-transitions 64 and ε-transitions 66. The automaton A_ε is k-closed for (R, +, *, 0, 1). Figure 6(b) shows the weighted automaton B 70 with states 72 and all non-ε-transitions 74 (only one transition is labeled) that is equivalent to automaton A 60. The general algorithm applies in this case since A_ε is acyclic.

As mentioned above with respect to the first step of the method according to the first embodiment of the invention, the computation of ε-closures is equivalent to

that of all-pairs shortest-distances over the semiring K in A_ε. There exists a generalization of the algorithm of Floyd-Warshall for computing the all-pairs shortest-distances over a semiring K under some general conditions. However, the running time complexity of that algorithm is cubic:

$$O(|Q|^3(T_\oplus + T_\otimes + T_*))$$

where $T_\oplus$, $T_\otimes$ and T_* denote the cost of $\oplus$, $\otimes$, and closure operations in the semiring considered. The algorithm can be improved by first decomposing A_ε into its strongly connected components, and then computing all-pairs shortest distances in each components visited in reverse topological order. However, it is still impractical for large automata when A_ε has large cycles of several thousand states. The quadratic space complexity $O(|Q|^2)$ of the algorithm also makes it prohibitive for such large automata. Another problem with this generalization of the algorithm of Floyd-Warshall is that it does not exploit the sparseness of the input automaton.

There exists a generic single-source shortest-distance algorithm that works with any semiring covered by the framework. The algorithm is a generalization of the classical shortest-paths algorithms to the case of the semirings of this framework.

This generalization is not trivial and does not require the semiring to be idempotent. In particular, a straightforward extension of the classical algorithms based on a relaxation technique would not produce the correct result in general. The algorithm is also generic in the sense that it works with any queue discipline. Figure 7 illustrates the pseudocode of the algorithm. The algorithm of Figure 7 works in particular with the semiring (R, +, *, 0, 1) when the weight of each cycle of A_ε admits a well-defined closure.

Referring to Figure 7, a queue S is used to maintain the set of states whose

leaving transitions are to be relaxed. S is initialized to $\{s\}$. For each state $q \in Q$, two attributes are maintained: $d[q] \in K$ an estimate of the shortest distance from s to q, and $r[q] \in K$ the total weight added to $d[q]$ since the last time q was extracted from S. Lines 1 — 3 initialize arrays d and r. After initialization, $d[q]$ $r[q] = \bar{O}$ for $q \in Q$ — $\{s\}$, and $d[s] = r[s] = \bar{1}$.

Given a state $q \in Q$ and a transition $e \in E[q]$, a relaxation step on e is performed by lines 11-13 of the pseudocode, where r is the value of $r[q]$ just after the latest extraction of q from S if q has ever been extracted from S, its initialization value otherwise.

Each time through the while loop of lines 5-15, a state q is extracted from S (lines 6-7). The value of $r[q]$ just after extraction of q is stored in r, and then $r[q]$ is set to $\bar{O}$ (lines 8-9). Lines 11-13 relax each transition leaving q. If the tentative shortest distance $d[n[e]]$ is updated during the relaxation and if $n[e]$ is not already in S, the state $n[e]$ is inserted in S so that its leaving transitions be later relaxed (lines 14-15). Whenever $d[n[e]]$ is updated, $r[n[e]]$ is updated as well. $r[n[e]]$ stores the total weight $\oplus$ added to $d[n[e]]$ since $n[e]$ was last extracted from S or since the time after initialization if $n[e]$ has never been extracted from S. Finally, line 16 resets the value of $d[s]$ to $\bar{1}$.

In the general case, the complexity of the algorithm depends on the semiring considered and the queue discipline chosen for S:

$$O(|Q| + (T_{\oplus} + T_{\otimes} + C(A))|E| \max_{q \in Q} N(q) + (C(I) + C(E))\Sigma_{q \in Q} N(q))$$

where $N(q)$ denotes the number of times state q is extracted from S, $C(E)$ the worst cost of removing a state q from the queue S, $C(I)$ that of inserting q in S, and $C(A)$ the cost of an assignment. This includes the potential cost of reorganization

of the queue to perform this assignment.

In the case of the tropical semiring $(R_+ \cup \{oo\}, mm, +, oo, 0)$, the method shown in Figure 7 coincides with classical single-source shortest-paths algorithms. In particular, it coincides with Bellman-Ford's algorithm when a FIFO queue discipline is used and with Dijkstra's algorithm when a shortest-first queue discipline is used. Using Fibonacci heaps, the complexity of Dijkstra's algorithm in the tropical semiring is:

$$O(|E| + |Q| \log |Q|)$$

The book Introduction to Algorithms by Cormen, Leiserson and Rivest, referenced above, provides further material for these algorithms. The complexity of the algorithm is linear in the case of an acyclic automaton with a topological order queue discipline:

$$O(|Q| + (T_\oplus + T_\otimes)|E|)$$

Note that the topological order queue discipline can be generalized to the case of non-acyclic automata A_ε, by decomposing A_ε, into its strongly connected components. Any queue discipline can then be used to compute the all-pairs shortest distances within each strongly connected component.

The all-pairs shortest-distances of A_ε can be computed by running $|Q|$ times the generic single-source shortest-distance algorithm. Thus, when A_ε is acyclic, that is, when A_ε admits no ε-cycle, then the all-pairs shortest distances can be computed in quadratic time:

$$O(|Q^2| + (T_\oplus + T_\otimes)|Q|\,|E|)$$

When A_ε is acyclic, the complexity of the computation of the all-pairs shortest distances can be substantially improved if the states of A_ε are visited in reverse

topological order and if the shortest-distance algorithm is interleaved with the actual removal of ε's. Indeed, one can proceed in the following way for each state p of A_ε visited in reverse topological order: (1) Run a single-source shortest-distance algorithm with source p to compute the distance from p to each state q reachable from p by ε's; and (2) Remove the ε-transitions leaving q as described in the previous section.

The reverse topological order guarantees that the ε-paths leaving p are reduced to the ε-transitions leaving p. Thus, the cost of the shortest-distance algorithm run from p only depends on the number of ε-transitions leaving p and the total cost of the computation of the shortest-distances is linear:

$$O(|Q| + (T_\oplus + T_\otimes)|E|)$$

In the case of the tropical semiring and using Fibonacci heaps, the complexity of the first stage of the algorithm is:

$$O(|Q|\cdot|E| + |Q|^2 \log |Q|)$$

In the worst case, in the second stage of the algorithm each state q belongs to the ε-closure of each state p, and the removal of ε's can create in the order of $|E|$ transitions at each state. Hence, the complexity of the second stage of the algorithm is:

$$O(|Q|^2 + |Q|\cdot|E|)$$

Thus, the total complexity of the algorithm in the case of an acyclic automaton A_ε is:

$$O(|Q|^2 + (T_\oplus + T_\otimes |Q|\cdot|E|)$$

In the case of the tropical semiring and with a non acyclic automaton A_ε the total complexity of the algorithm is:

$$O(|Q|\cdot|E| + |Q|^2 \log |Q|)$$

For some automata of about a thousand states with large ε-cycles, the inventor of the present invention has implemented the automata with an improvement of up to 600 times faster than the previous implementation based on a generalization of the Floyd-Warshall algorithm.

An important feature of the system and method of the present invention is that it admits a natural on-the-fly implementation. Indeed, the outgoing transitions of state q of the output automaton can be computed directly using the ε-closure of q. However, with an on-the-fly implementation, a topological order cannot be used for the queue S even if A_ε is acyclic since this is not known ahead of time. Thus, both an off-line and an on-the-fly version of the system and method of the present invention may be implemented. An on-the-fly algorithm can save space by removing only those epsilons that are necessary for the particular use of the input machine. An off-line algorithm simply applies to the entire machine and constructs the output regardless of what portion was really needed. In some cases as discussed above, the complexity of the off-line version is better because it can take advantage of the specific property of the machine (the fact that it is acyclic for example). This is not possible in the on-the-fly case. To allow for both uses, the present disclosure covers both the on-the-fly implementation and the off-line context.

The shortest-distance algorithm presented herein admits an approximation version where the equality of line 11 in the pseudocode of Figure 7 is replaced by an approximate equality modulo some predefined constant δ. This can be used to remove ε-transitions of a weighted automaton A over a semiring K such as (R, +, *, 0, 1), even when K is not k-closed for A_ε. Although the transition weights are then only approximations of the correct results, this may be satisfactory for many practical

purposes such as speech processing applications. Furthermore, one can arbitrarily improve the quality of the approximation by reducing δ. As mentioned before, one can also use a generalization of the Floyd-Warshall algorithm to compute the result in an exact way in the case of machines A_ε with relatively small strongly connected components and when the weight of each cycle of A_ε admits a well-defined closure.

In another aspect of the first embodiment of the invention, a similar removal method is applied to transducers rather than automatons. The method according to this aspect of the first embodiment of the invention is closely related to the ε-removal method described above. As shown in Figures 8(a), let T_1 80 be a weighted transducer over a k-closed semiring K. T_1 includes states 82 and transitions with input label ε 84. In some applications, one may wish to construct a new transducer T_2 equivalent to T_1 such that T_2 has no ε-transition and such that along any successful path of T_2, no transition with input label different from c is preceded with a transition with input label ε. An exemplary transducer T_2 90 is shown in Figure 8(b) with states 92 and transitions 94 having the appropriate ε-transition removes as described above. Figure 8(b) is the output of epsilon normalization. You don't remove all epsilon translation.

A transducer such as T_2 90 is input-normalized and a method for constructing T_2 90 from T_1 80 is called an input ε-normalization method. The advantage of using a T_2 90 so constructed is similar to the benefits of the ε-removal method of the first aspect of the first embodiment of the invention. Matching its input with algorithms such as composition does not require any unnecessary delay due to the presence of input ε's. Figures 8(a) and 8(b) illustrate the application of the algorithm in the case of a weighted transducer over the tropical semiring.

Note that the system (Σ^*, $\cup$, $\cap$, $\emptyset$, Σ^*) defines an idempotent semiring K'

over the set of strings Σ^*. A weighted transducer T_1 80 over the semiring K can be viewed as a weighted automaton A over the cross-product semiring S = K' x K. An element of S is a pair (χ, w) where $\chi \in \Sigma^*$ is a string and $w \in K_1$ the original *weight* of the transducer.

The input ε-normalization method applies only to weighted transducers T_1 80 over an arbitrary semiring K that do not admit any cycle with input label ε. Cycles with both input and output can be removed using for example the ε-removal method described according to the first aspect of the first embodiment of the invention. Since T_1 80 does not admit any cycle with input label c, A does not admit any ε-cycle and the semiring S is k-closed for A_ε. Thus the shortest-distance algorithm presented above can be used with A_ε and, more generally, the ε-removal algorithm presented in the previous section applies to A.

This input ε-normalization system and method relates to the ε-removal method in the specific case of semirings of the S type. The result of the ε-removal method is an acceptor with no ε but with weights in S. Figure 9(a) shows the result of that algorithm when applied to the transducer T_1 of Figure 8(a). The transducer 100 of Figure 9(a) includes states 102 and transitions 104 (not all labeled) without any ε-transitions. The representation of transducers does not allow output labels to be strings, but transitions can be replaced with output strings by a set of consecutive transitions with outputs in $\Sigma \cup \{\varepsilon\}$ in a trivial fashion. The result of that transformation is of interest and is useful in many contexts. However, the resulting transducer does not have the normalization property described above. To ensure that input ε's are not found on a path before an input non-ε label, there is a need to normalize the result of ε-removal in a different way. To each state p, a residual output

string y is associated that needs to be output from that state.

States in the result of ε-normalization correspond to pairs (p, y) where p is a state of the original machine and y a residual output string. The residual string associated to an initial state is just ε. The outgoing transitions of a state (p, y_0) are determined as follows. For each transition from state p to q with input label χ, output string y_1 and weight w_1 in the result of the ε-removal algorithm, a transition is created from state (p, y_0) to state $(q, a^{-1}y_0y_1)$, with input χ, output a and weight w_1, where a is the first letter of y_0y_1, $a=\varepsilon$ *if* $y_0y_1 \varepsilon$. A state (p,y_0) is final if p is a final state of the original machine and if $y_0 = \varepsilon$. When p is final and $y_0 \neq \varepsilon$, then a transition is created from (p, y_0) to (p, y_i) with input ε and output a with $y_0 ay_1$.

Figure 9 (b) illustrates this construction for the particular case of the transducer of Figure 8 (a). As in the case of ε-removal, input c-normalization admits a natural on-the-fly implementation since the outgoing transitions of state (p, y_0) 112 of the output automaton can be computed directly using only y_0 and the input ε-closure of q. The transducer 110 of Figure 9(b) illustrates states 112 and transitions 114.

The method according to the first embodiment of the invention may be implemented in numerous ways. For example, high level programming languages such as Basic, C or other programming language may be used to implement the method. The best mode of implementation is a software implementation using a high-level programming language.

Second embodiment of the invention

The second embodiment of the invention relates to a system or apparatus for ε-removal in automata. The system may comprise a variety of different forms. For example, a preferable aspect of the second embodiment of the invention is to provide a hardware implementation using electronic circuitry such as an EPROM, a programmable logic device (PLD) or a large group of logic gates. Some design tools such as SPLat (Simple Programmed Logic automation tool) boards have specifically been designed for implementation and programming of FSMs. Thus, circuit boards may be designed and created to carry out the functionality of the methods disclosed as part of the first embodiment of the invention. According to this embodiment of the invention, a hardware implementation such as a computer circuit, EPROM, or logic board is programmed to perform the steps set forth in the first embodiment of the invention. In this regard, the hardware, in whatever form, will receive an input weighted automaton A with ε-transitions. Upon the operation of the hardware component to the automaton A with ε-transitions, an equivalent weighted automaton B with no ε-transitions is created.

The process performed by the hardware comprises two main steps. The first step consists of computing for each state "p" of the input automaton A its ε-closure denoted by C[p]:

$$C[p] = \{(q,w): q \in \varepsilon[p], d[p,q] = w \in K - \{\bar{O}\}\}$$

Where $\varepsilon[p]$ represents a set of states reachable from "p" via a path labeled with ε. The second step consists of modifying the outgoing transitions of each state "p" by removing those labels with ε and by adding to be non-ε-transitions leading each state q with their weights pre-$\otimes$-multiplied by $d[p,q]$.

The second step in the method comprises modifying the outgoing transitions

of each state "p" by removing those labeled with ε. The method adds to the set of transitions leaving the state "p" non-ε-transitions leaving each state "q" in the set of states reachable from "p" via a path labeled with ε with their weights pre-$\otimes$-multiplied by the ε-distance from state "p" to state "q" in the automaton A. State "p" is a final state if some state "q" within the set of states reachable from "p" via a path labeled with ε is final and the final weight ρ [p] is $\rho[p] = \bigoplus_{q \in e[p] \cap F} (d[p,q] \otimes \rho[q])$.

One aspect of the second embodiment of the invention relates to a computer readable medium such as a data disc or other medium having a program stored thereon that is operable to perform the method or process described with respect to the first embodiment of the invention. In this regard, executable code or source code may be stored on the computer readable medium. Those of ordinary skill in the art will understand and recognize how to create such a computer readable medium according to this embodiment of the invention given the understanding of the method according to the first embodiment of the invention. The computer readable medium may be, for example, a computer disk, hard disk, read-only-memory, or random access memory.

Third embodiment of the invention

The third embodiment of the invention relates to product produced by the process described above with respect to the method explained above. Specifically, the equivalent weighted automaton B with no ε-transitions is a product that is created by the process set forth and described above as the first embodiment of the invention. Therefore, the third embodiment of the invention relates to an automaton having no ε-transitions, the automaton being created according to the ε-removal algorithms described herein.

What follows is a series of equations related to process of Generic ϵ -Removal and Input ϵ -Normalization Algorithms for Weighted Transducers. We begin with a theorem and then the proof of the theorem. The following theorem and proof provide disclosure related to another aspect of the present invention.

Theorem 1 Let $A = (\Sigma, Q, I, F, E, \lambda, p)$ be a weighted automaton over the semiring K right k-closed for A. Then the weighted automaton B result of the ε-removal algorithm just described is equivalent to A.

Proof. We show that the function defined by the weighted automaton A is not modified by the application of one step of the loop of the pseudocode above. Let $A = (\Sigma, Q, I, F, E, \lambda, p)$ be the automaton just before the removal of the ϵ-transitions leaving state $p \in Q$.

Let $\chi \in \Sigma^*$, and let $Q(p)$ denote the set of successful paths labeled with χ passing through p and either ending at p or following an ε-transition of p. By commutativity of $\oplus$, we have:

$$[A](\chi) = \bigoplus_{\pi \in P(x) - Q(p)} \lambda(p[\pi]) \otimes \omega[\pi] \otimes \rho(n[\pi]) \oplus \bigoplus_{\pi \in Q(p)} \lambda(\rho[\pi]) \otimes \omega[\pi] \otimes \rho(\eta[\pi])$$

Denoted by S_1 and S_2 are the first and second term of that sum. The ε-removal at state p does not affect the paths not in $Q(p)$, thus we can limit our attention to the second term S_2. A path in $Q(p)$ can be factored in: $\pi = \pi_1, \pi_2, \pi_3$ where π_ϵ, is a portion of the path from p to $n\pi_\epsilon = q$ labeled with ϵ. The distributivity of $\otimes$ over $\oplus$ gives us:

$$S_2 = (\bigoplus_{n[\pi_1]=\rho} \lambda(p[\pi_1]) \otimes \omega[\pi_1])) \otimes S_{22}$$

with:

$$S_{22} = (\bigoplus_{p[\pi_\varepsilon]=p, n[\pi_\varepsilon]=q} \omega[\pi_e]) \otimes \bigoplus_{\rho[\pi_2]=q} \omega[\pi_2] \otimes \rho(n[\pi_2])$$

$$= d[p,q] \otimes \bigoplus_{p[\pi_2]=q} \omega[\pi_2] \otimes \rho(n[\pi_2])$$

$$= \bigoplus_{\rho[\pi_2]=q} (d[p,q] \otimes \omega[\pi_2] \otimes \rho(n[\pi_2])$$

For paths π such that $\pi_2 = \epsilon$:

$$S_{22} = \bigoplus_{p[\pi_2]=q, q \in \varepsilon[p]} (d[p,q] \otimes \rho(q))$$

which is exactly the final weight associated to p by ε-removal at p. Otherwise, by definition of π_ϵ, π_2 does not start with an ε-transition. The second term of the sum defining S_{22} can thus be rewritten as:

$$S_{22} = \bigoplus_{e \in E[q], i[e] \neq \epsilon, \pi_2 = e\pi_2'} (d[p,q] \otimes \omega[e]) \otimes \omega[\pi_2'] \otimes \rho(n[\pi_2])$$

The ε-removal step at state p follows exactly that decomposition. It adds non ε-transitions e leaving q with weights *(d[p, q] ⊗ w[e])* to the transitions leaving p. This ends the proof of the theorem.

Although the above description may contain specific details, they should not be construed as limiting the claims in any way. Other configurations of the described embodiments of the invention are part of the scope of this invention. For example, finite state machines may be used to design computer programs, logic circuits, or electronic control systems. The ε-removal algorithm and other algorithms disclosed herein may clearly have application outside of speech processing. For example, the same algorithm (epsilon-removal) applies in contexts similar to speech such as text processing, information extraction, information retrieval, computational biology, etc. where weighted automata are used and where one wishes to make their use more efficient. Furthermore, the system and method disclosed here can be straightforwardly modified to remove transitions with a label a different from ε. This

can be done for example by replacing ε by a new label and a by ε, applying ε-removal and then restoring original ε's. Accordingly, the appended claims and their legal equivalents should only define the invention, rather than any specific examples given.